Predicting and Controlling Resource Usage in an Active Network

Stephen F. Bush and Amit B. Kulkarni (GE CRD*)
Virginie Galtier, Yannick Carlinet and Kevin L. Mills (NIST)
Livio Ricciulli (Metanetworks)

DARPA Active Networks PI Meeting December 6–9, 2000

***with assistance from Scott S. Shyne (AFRL), COTR for GE CRD contract**

Demonstration Team

General Electric Corporate Research & Development

Stephen F. Bush, Active Virtual Network Management Prediction
Amit B. Kulkarni, Magician EE and Active Applications

NIST Information Technology Laboratory

Virginie Galtier, Active Application Modeling and Measurement
Yannick Carlinet, Active Node Calibration
Kevin L. Mills, Principal Investigator
Stefan D. Leigh, Statistical Data Analysis
Andrew Rukhin, Statistical Model Design

Metanetworks

Livio Ricciulli, Active Network Management Interface Design

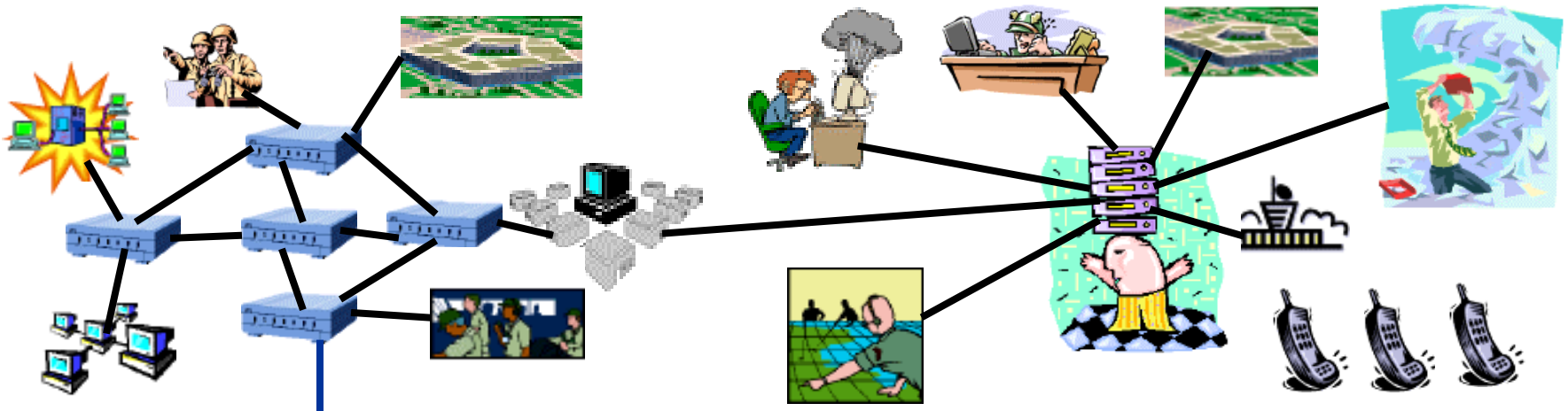
Presentation Outline

- Relevance of the Demonstrated Technology
- Integration Requirements for the Demonstration
- Details about the Technologies underlying the Demonstration
- Demonstrations
  - *#1 Detect and Kill Malicious or Erroneous Packets*
  - *#2 Demonstrate the predictive power of AVNMP when combined with NIST CPU usage prediction models*
- Accomplishments and Lessons Learned
- Future Research

Relevance of the Technology

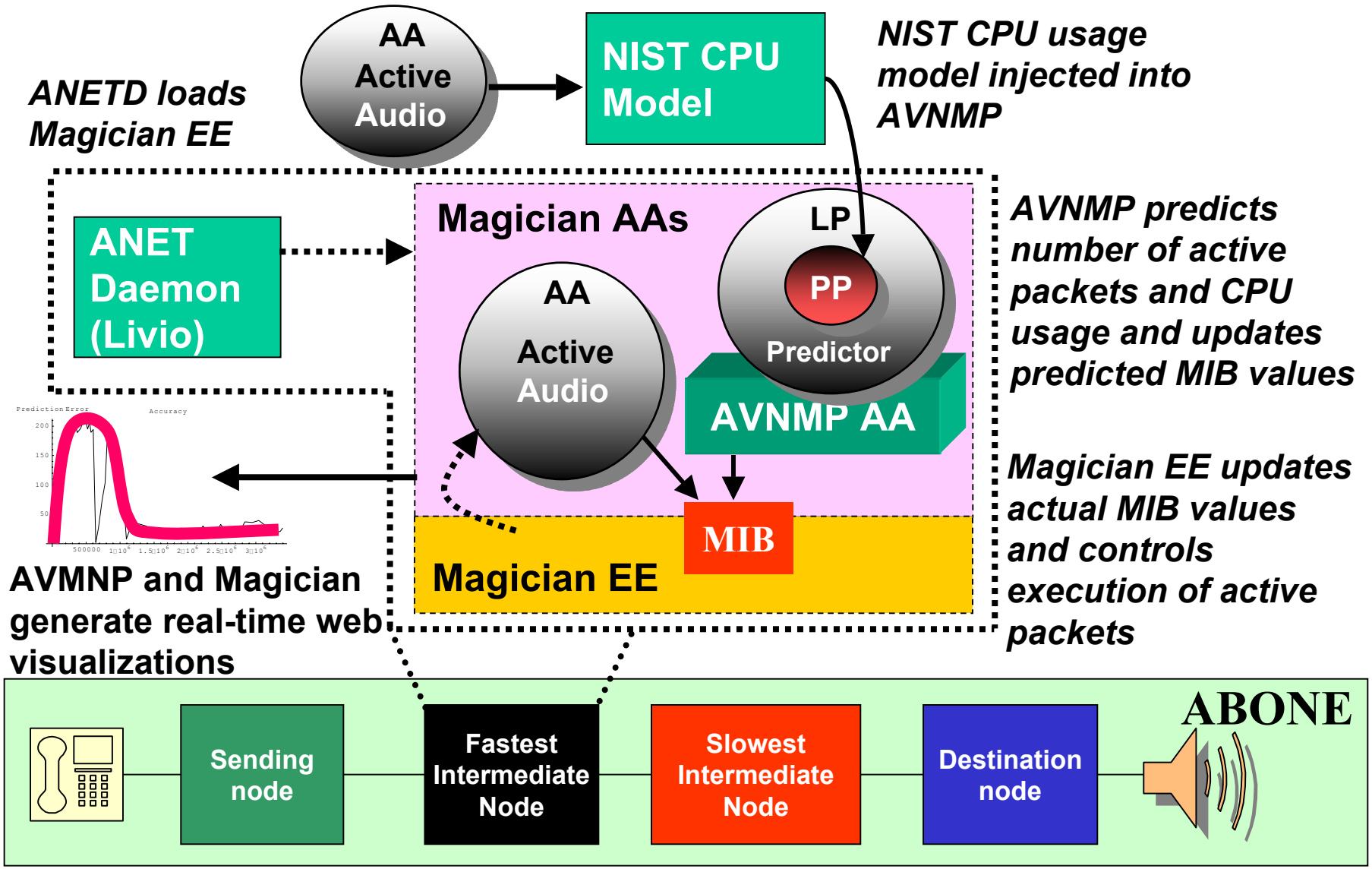
FAULT RESILIENCY

OVERLOAD PREDICTION



INTEROPERABLE MANAGEMENT OF HETEROGENEOUS RESOURCES

Integration Requirements

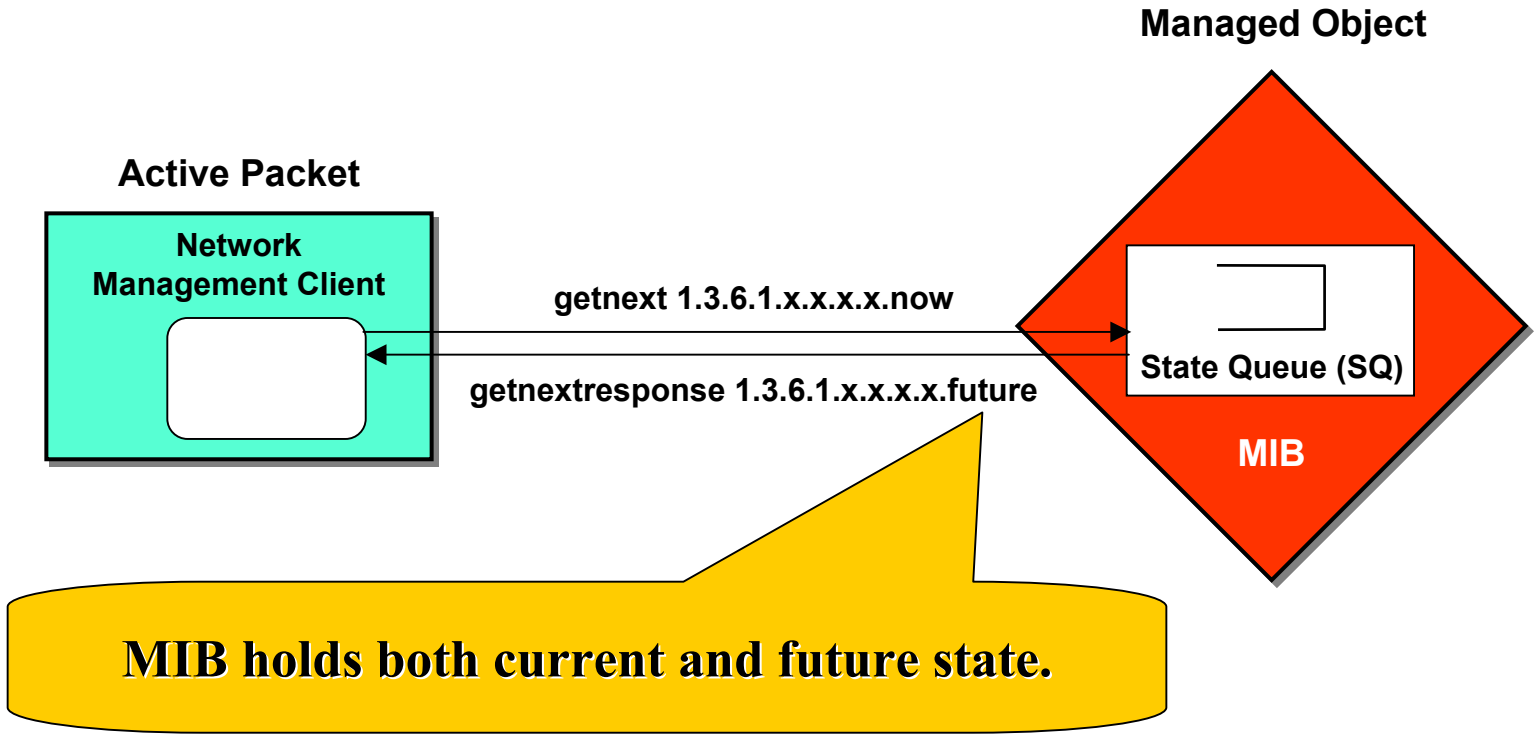


What's Ahead

- What is AVNMP and how does it work? (Steve Bush)
- How is AVNMP integrated with the Magician EE? (Amit Kulkarni)
- How does NIST model CPU usage? (Kevin Mills)
- How are NIST CPU models integrated with Magician? (Amit Kulkarni)
- Does this integrated technology work?
  - *#1 Detect and Kill Malicious or Erroneous Packets* (Amit Kulkarni)
  - *#2 Demonstrate the predictive power of AVNMP when combined with NIST CPU usage prediction models* (Steve Bush)
- What was accomplished and what lessons were learned? (Kevin Mills)
- What are some ideas for future research? (Kevin Mills and Steve Bush)

Self prediction

Communication networks that can predict their own behavior!

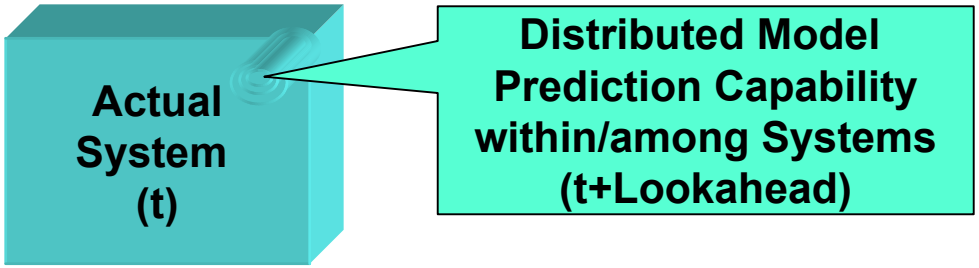


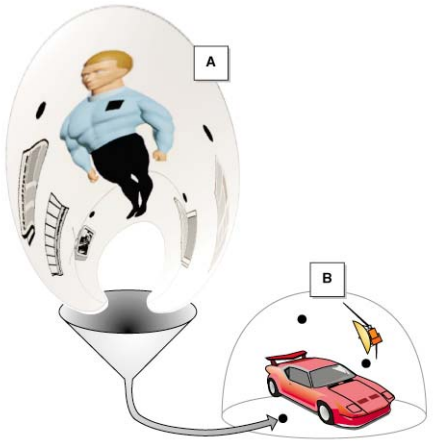
Some Uses for Self Prediction

- Optimal management polling interval is determined based upon predicted rate of change and fault probability
- Fault correction will occur before system is impacted
- Time to perform dynamic optimization of repair parts, service, and solution entity (such as software agent or human user) co-ordination
- Optimal resource allocation and planning
- "What-if" scenarios are an integral part of the network
- AVNMP-enhanced components protect themselves by taking action, such as migrating to "safe" hardware **before** disaster occurs

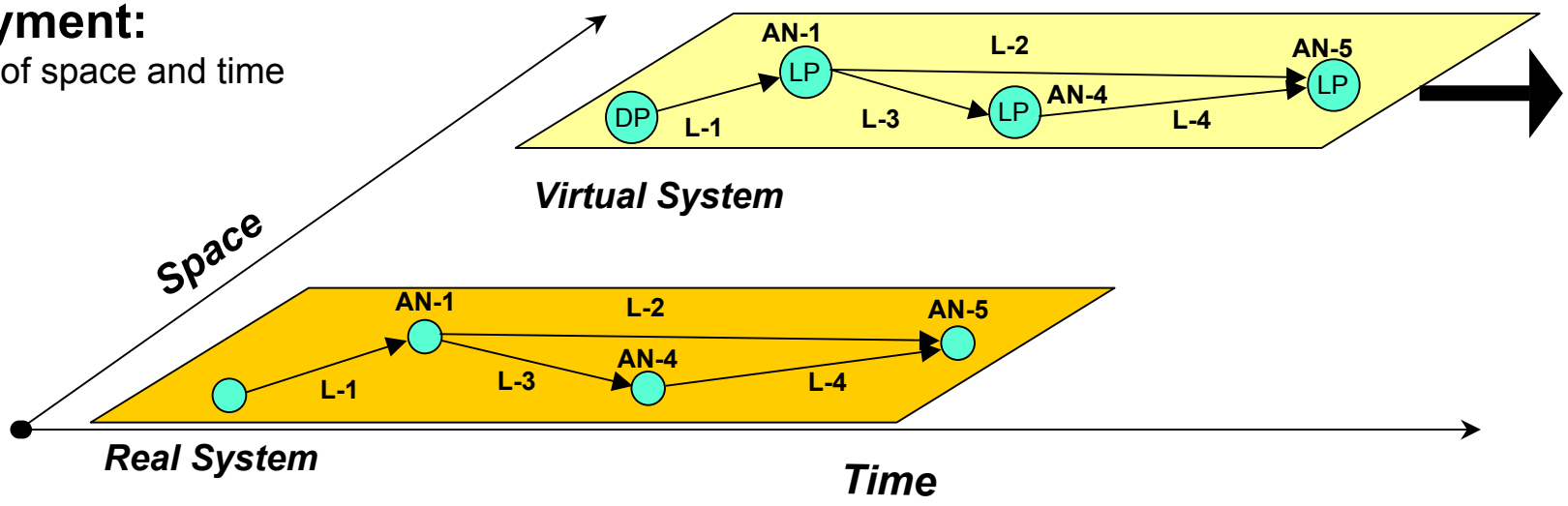
Injecting a Model into the Net

Goal: Active Virtual Network Management Prediction



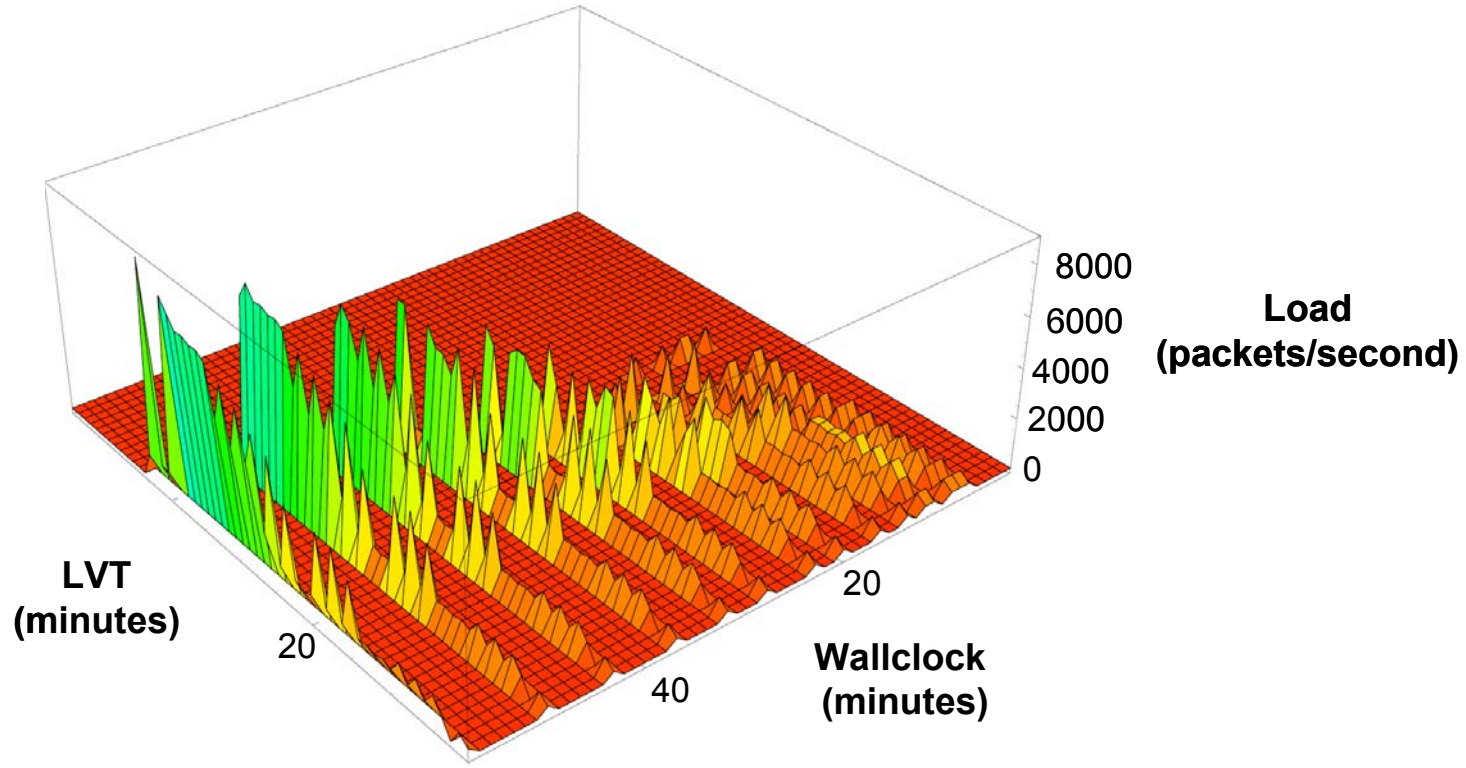


Deployment:
Best use of space and time



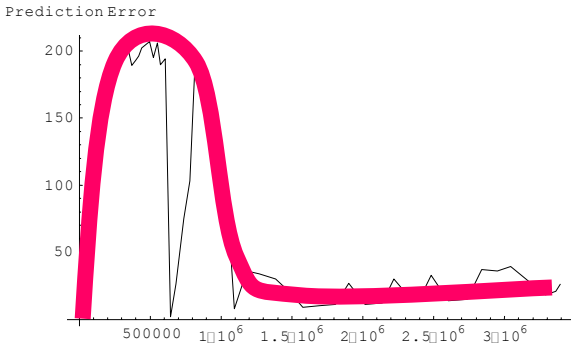
Cyclic Prediction Refinement

- Prediction ends when preset look ahead is reached
- Previous predictions are refined as time progresses



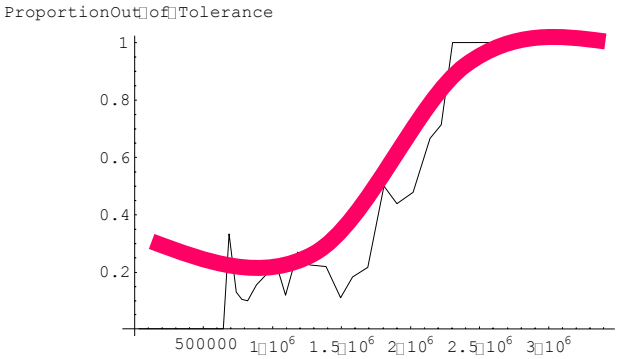
Accuracy-Performance Tradeoff

Prediction Error



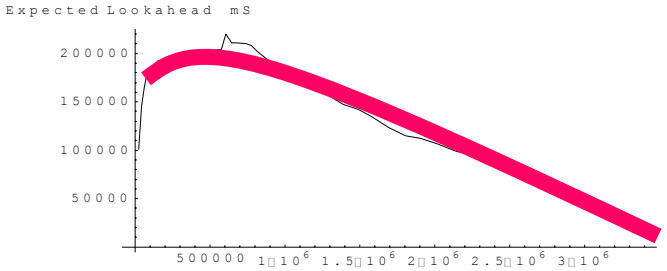
Experiment involved demanding more accuracy over time by reducing the error between predicted and actual values, however...

Out of Tolerance Messages



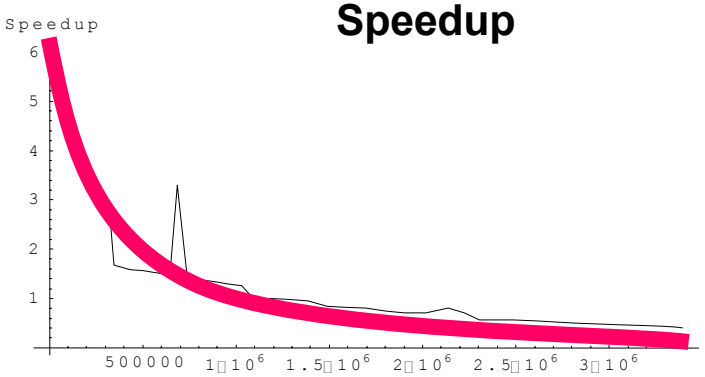
… this required more out-of-tolerance messages...

Look-ahead



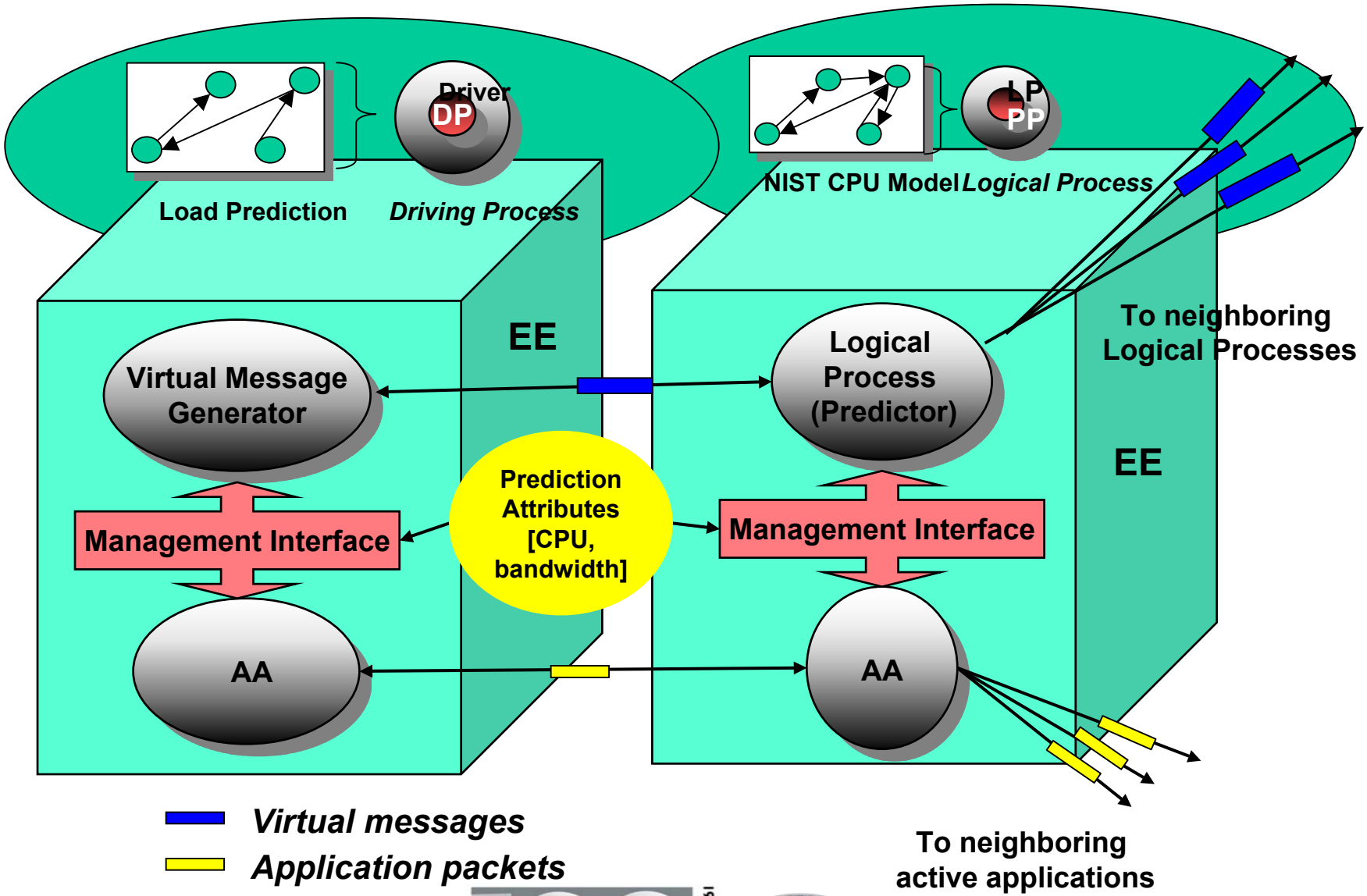
…the tradeoff was loss in Look-ahead...

Speedup



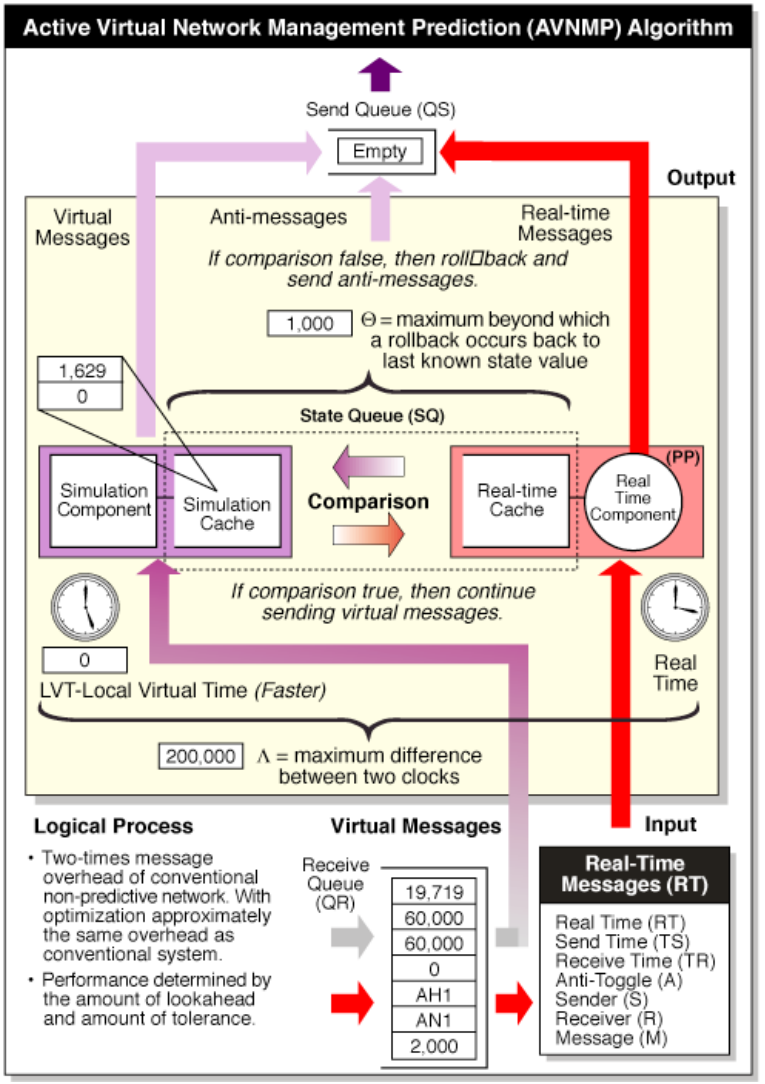
…. and loss in speedup

AVNMP Architecture

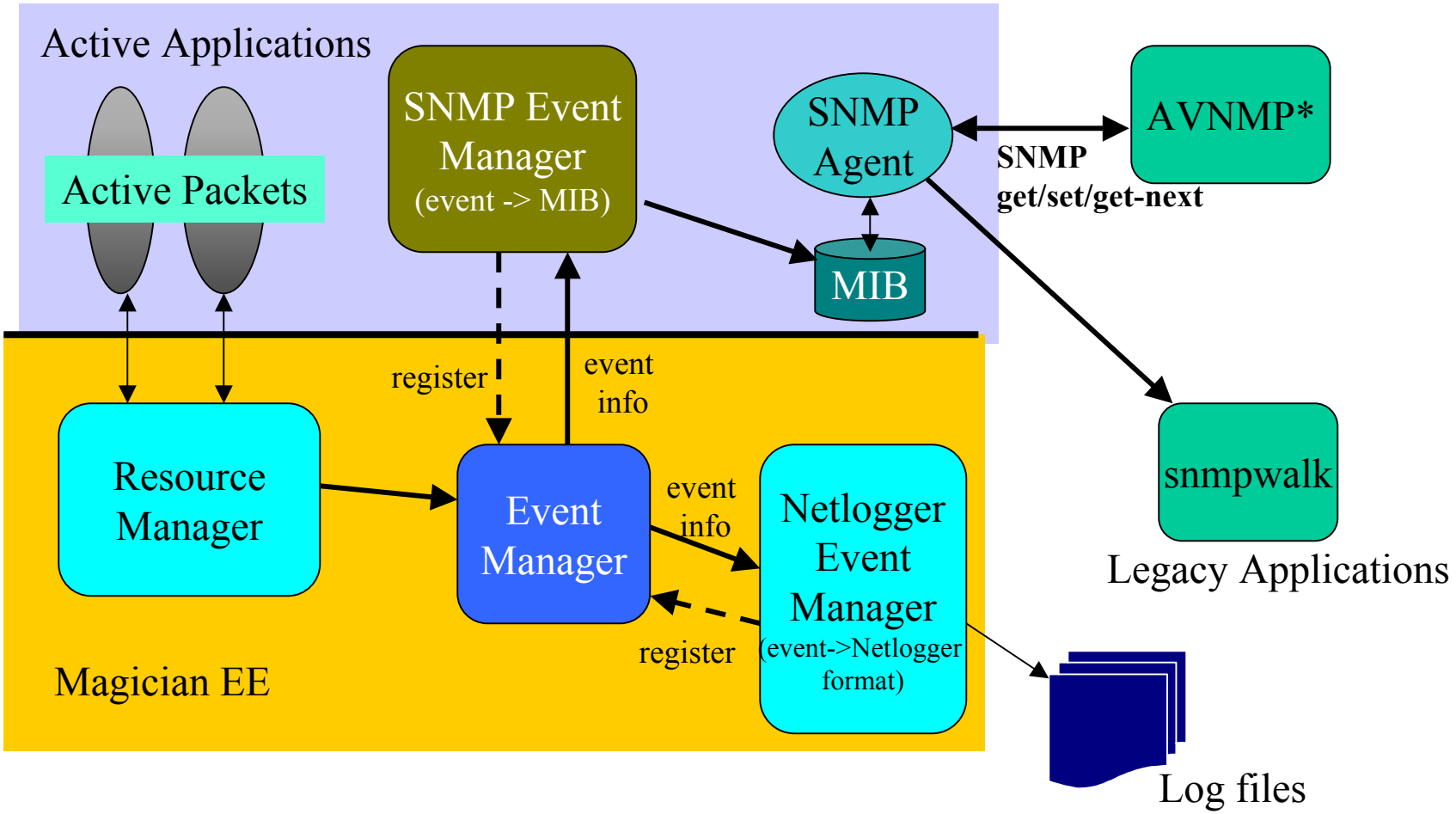


AVNMP Algorithm

- Prediction performance continuously kept within tolerance via rollback
- Time Warp-like technique used for maximum use of space and time in virtual system
- Rollback State Cache holds MIB future values
- Active Networks and Active Virtual Network Management Prediction: A Proactive Management Framework, Bush, Stephen F. and Kulkarni, Amit B. Kluwer Academic\Plenum Publishers. Spring 2001. ISBN 0-306-46560-4
- But how do AAs, such as AVNMP, communicate with each other, and with the EE? Two mechanisms:
  - Event reporting
  - SNMP communication

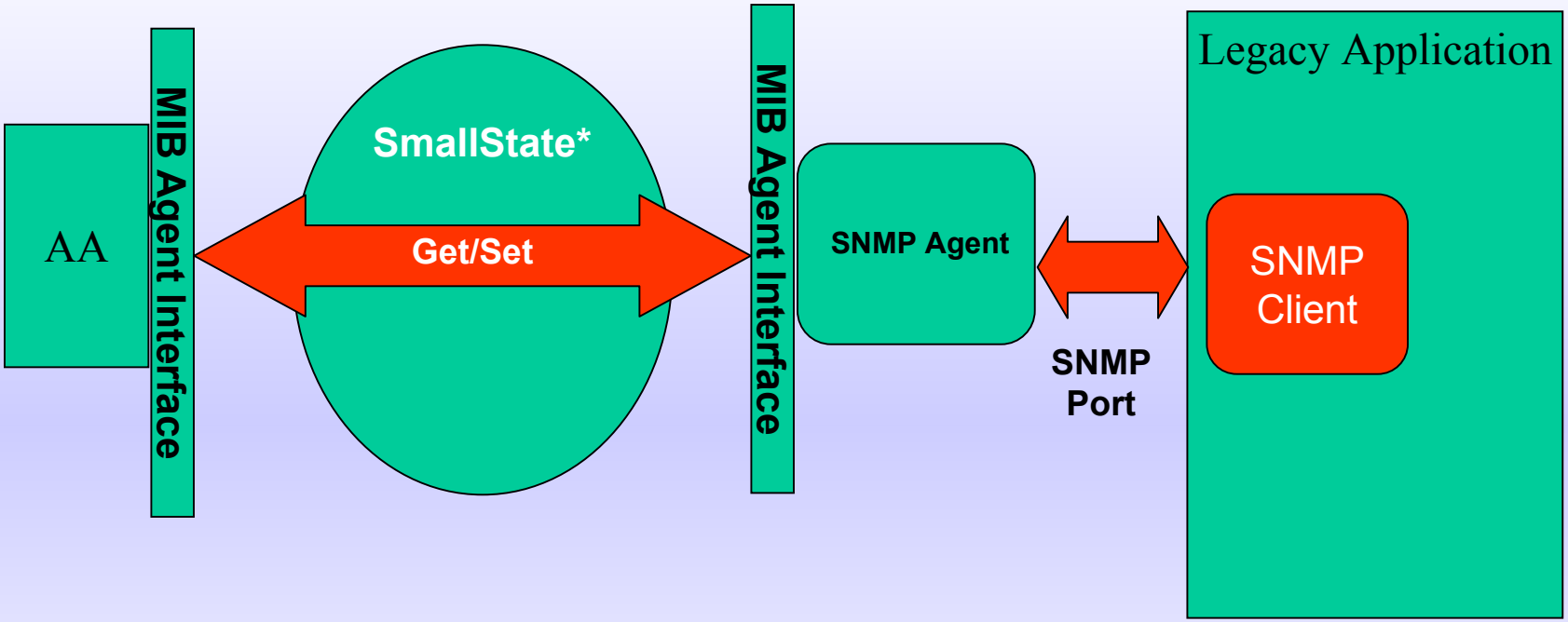


Magician Event Reporting Architecture



*note that AVNMP can run as a local or remote AA.

Active SNMP Interface

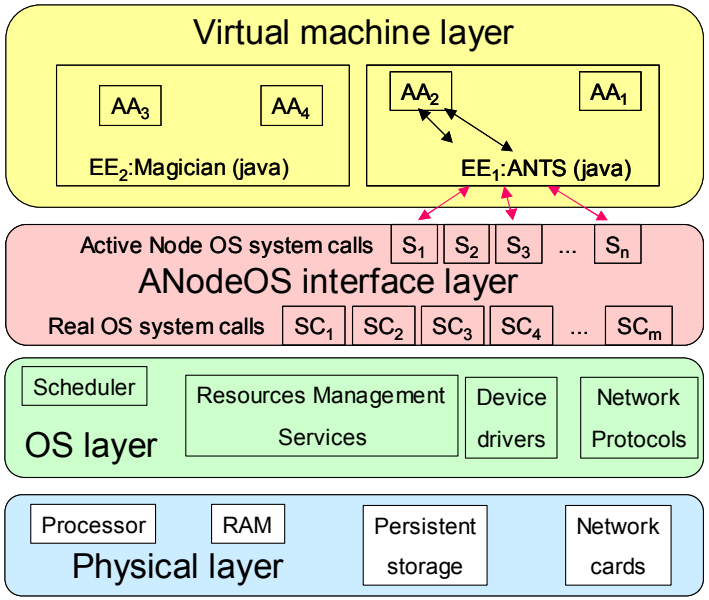


* Magician transient or soft state available to AAs

Overview of NIST Research

- Identified Sources of Variability Affecting CPU Time Use by Active Applications
- Developed a Mechanism for Monitoring and Measuring CPU Time Use by Active Applications
- Developed and Evaluated Models to Characterize CPU Use by Active Applications
- Developed and Evaluated a Technique to Scale Active Application Models for Interpretation among Heterogeneous Nodes

Sources of Variability



ANETS ARCHITECTURE

VARIABILITY IN EXECUTION ENVIRONMENT

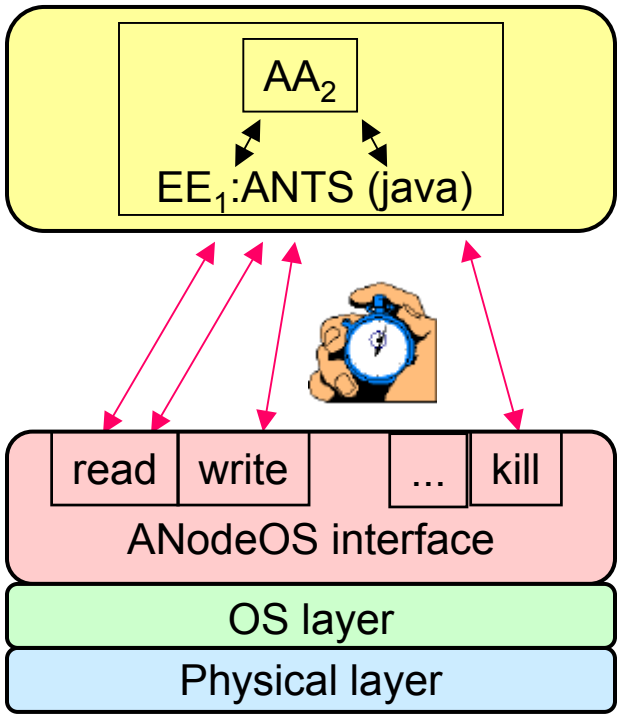
Trait	Blue	Black	Green
CPU Speed	450 MHz	333 MHz	199 MHz
Processor	Pentium II	Pentium II	PentiumPro
Memory	128 MB	128 MB	64 MB
OS	Linux 2.2.7	Linux 2.2.7	Linux 2.2.7
JVM	jdk 1.1.6	jdk 1.1.6	jdk 1.1.6
Benchmark			
Avg. CPU us	534	479	843
Avg. PCCs	240,269	159,412	167,830

	Blue		Black		Green	
System Call	pcc	us	pcc	us	pcc	us
read	19,321	43	12,362	37	12,606	63
write	22,609	50	14,394	43	12,362	62
socketcall	27,066	60	17,591	53	14,560	73
stat	22,800	51	14,731	44	12,042	61

VARIABILITY IN SYSTEM CALLS

Measuring AA Executions

Monitor at System Calls in Active Node OS → **Generate Execution Trace**



…
begin, user (4 cc), read (20 cc), user (18 cc), write(56 cc), user (5 cc), end

begin, user (2 cc), read (21 cc), user (18 cc), kill (6 cc), user (8 cc), end

begin, user (2 cc), read (15 cc), user (8 cc), kill (5 cc), user (9 cc), end

begin, user (5 cc), read (20 cc), user (18 cc), write(53 cc), user (5 cc), end

begin, user (2 cc), read (18 cc), user (17 cc), kill (20 cc), user (8 cc), end
…

Trace is a series of system calls and transitions stamped with CPU time use

Modeling AA Executions

Consume Execution Trace → **Generate Active Application Model**

…
begin, user (4 cc), read (20 cc), user (18 cc), write(56 cc), user (5 cc), end

begin, user (2 cc), read (21 cc), user (18 cc), kill (6 cc), user (8 cc), end

begin, user (2 cc), read (15 cc), user (8 cc), kill (5 cc), user (9 cc), end

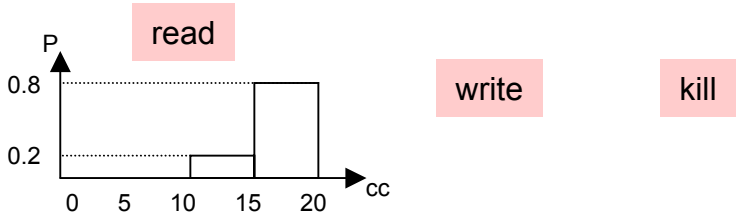
begin, user (5 cc), read (20 cc), user (18 cc), write(53 cc), user (5 cc), end

begin, user (2 cc), read (18 cc), user (17 cc), kill (20 cc), user (8 cc), end
…

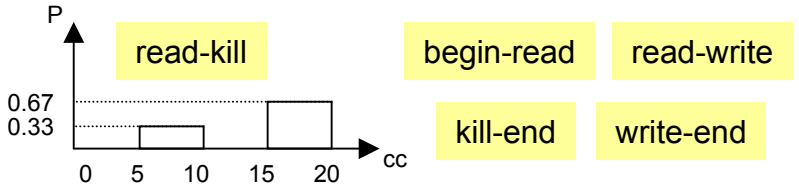
Scenario A:
sequence = "read-write",
probability = 2/5

Scenario B:
sequence = "read-kill",
probability = 3/5

Distributions of CPU time in system calls :



Distributions of CPU time between system calls :



Evaluating AA Models

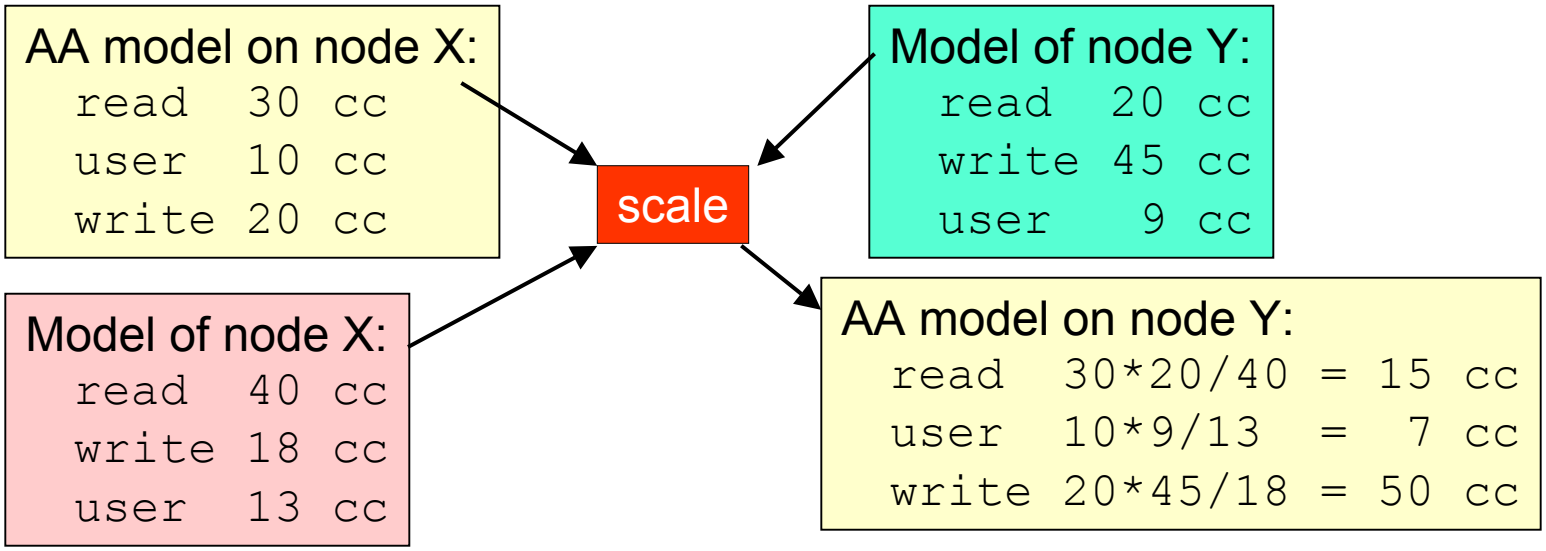
Simulate Model with Monte Carlo Experiment → **Statistically Compare Simulation Results against Measured Data**

		100 bins-20000 reps		50 bins-20000 reps		50 bins-500 reps	
EE	AA	Mean	Avg. High Per.	Mean	Avg. High Per.	Mean	Avg. High Per.
ANTS	Ping	0.86	0.9	0.64	2	2.70	10
	Mcast	0.40	1.9	0.35	3	4.91	16
Magician	Ping	0.44	33	0.70	32	1.77	32
	Route	0.73	13	0.30	12	6.66	23

The Average Absolute Deviation (in Percent) of Simulated Predictions from Measured Reality for Each of Two Active Applications in Two Different Execution Environments Running on One Node (Average High Percentile Considers Combined Comparison of 80th, 85th 90th, 95th, and 99th Percentiles) –Results Given for Models Composed Using Three Different Combinations of Bin Granularity (bins) and Simulation Repetitions (reps)

Scaling AA Models

- Each Node Constructs a Node Model using two benchmarks:
  - a system benchmark program ⌫ for each system call, average system time
  - for each EE, a user benchmark program ⌫ average time spent in the EE between system calls
- To scale an AA Model select one Node Model as a reference known by all other active nodes

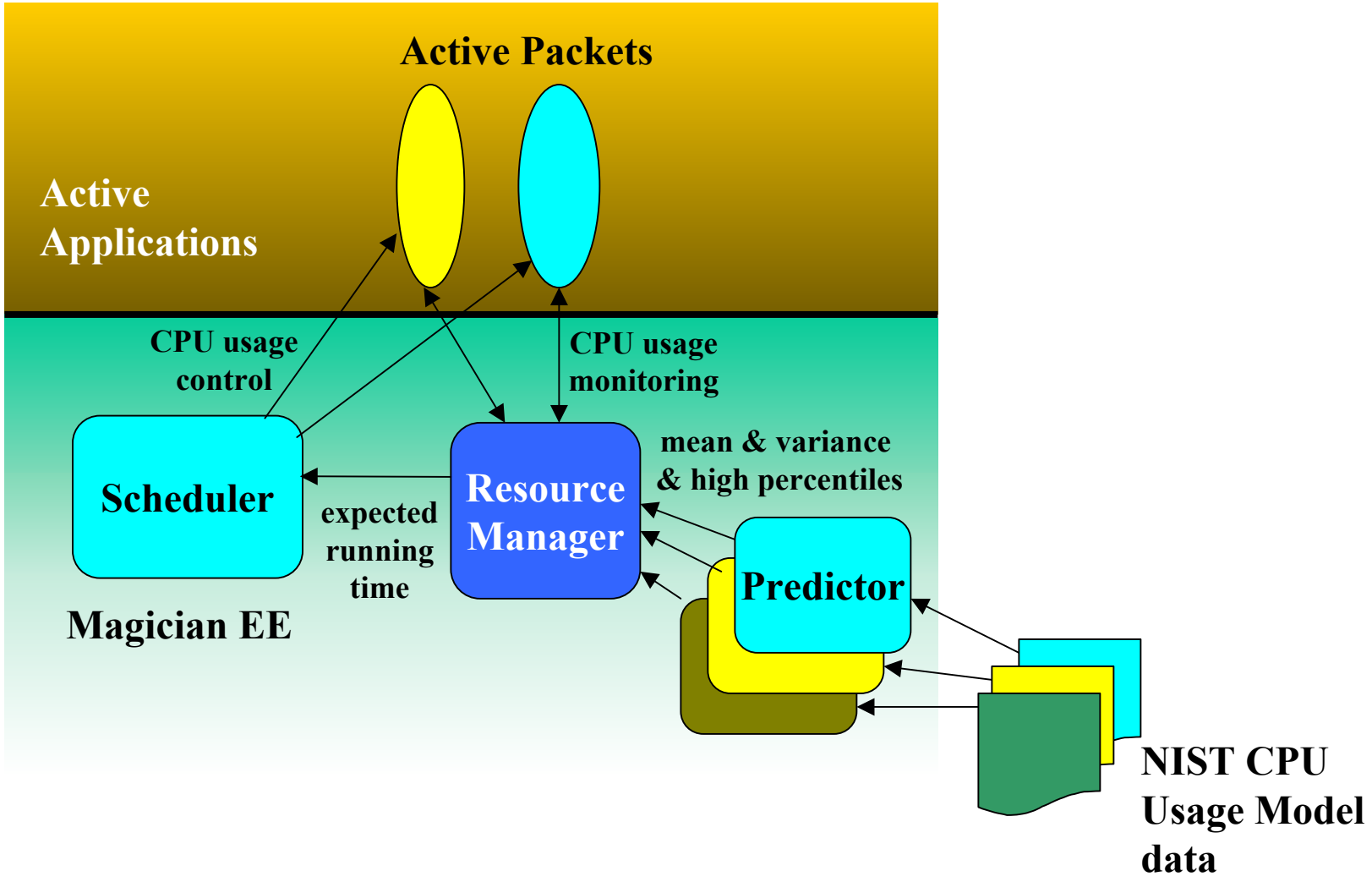


Evaluating Scaled AA Models

Prediction Error Measured when Scaling Application Models between Selected Pairs of Nodes vs. Scaling with Processor Speeds Alone

				Scaling with Models		Scaling with Speeds	
EE	AA	Node X	Node Y	Mean	Avg. High Per.	Mean	Avg. High Per.
ANTS	Ping	Black	Blue	2	4	12	14
		Black	Green	4	11	37	39
		Blue	Black	3	5	14	17
		Green	Blue	7	8	40	45
	Mcast	Black	Blue	0.3	7	10	13
		Blue	Black	3	11	11	16
		Green	Black	23	15	31	53
Magician	Ping	Blue	Black	4	49	60	58
		Blue	Green	3	36	48	47
		Black	Blue	3	18	37	36
	Route	Blue	Black	7	13	41	51
		Black	Blue	9	22	29	33
		Blue	Green	20	16	33	40

Implementing AA Models in Magician



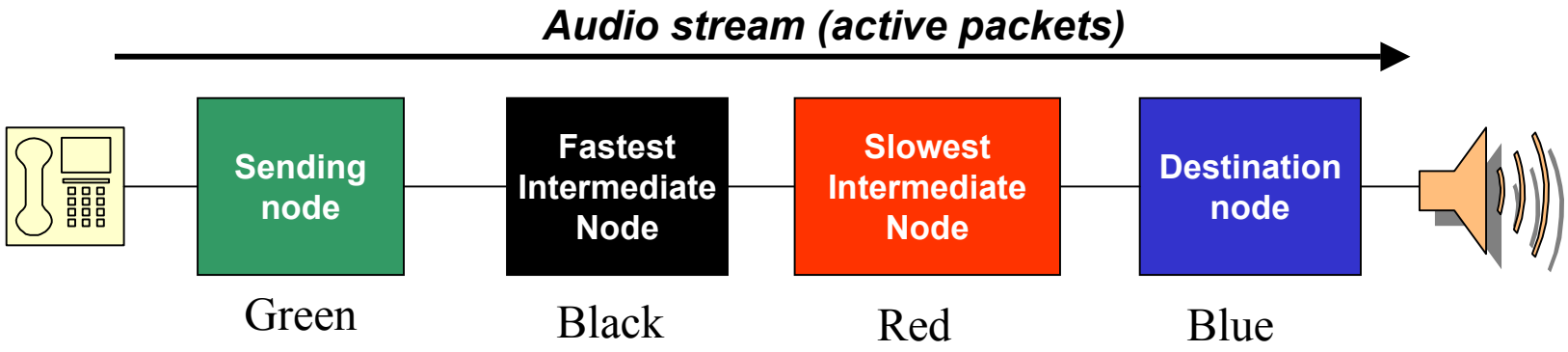
Demonstration #1 Overview

Detect and Kill Malicious or Erroneous Active Packets

- Illustrate motivation behind CPU usage modeling
- Compare three policies to enforce limits on CPU consumption
- Show improvement of NIST CPU usage models over naïve scaling (which is based solely on relative processor speeds)

Topology for Demonstration #1

Detect and Kill Malicious or Erroneous Active Packets



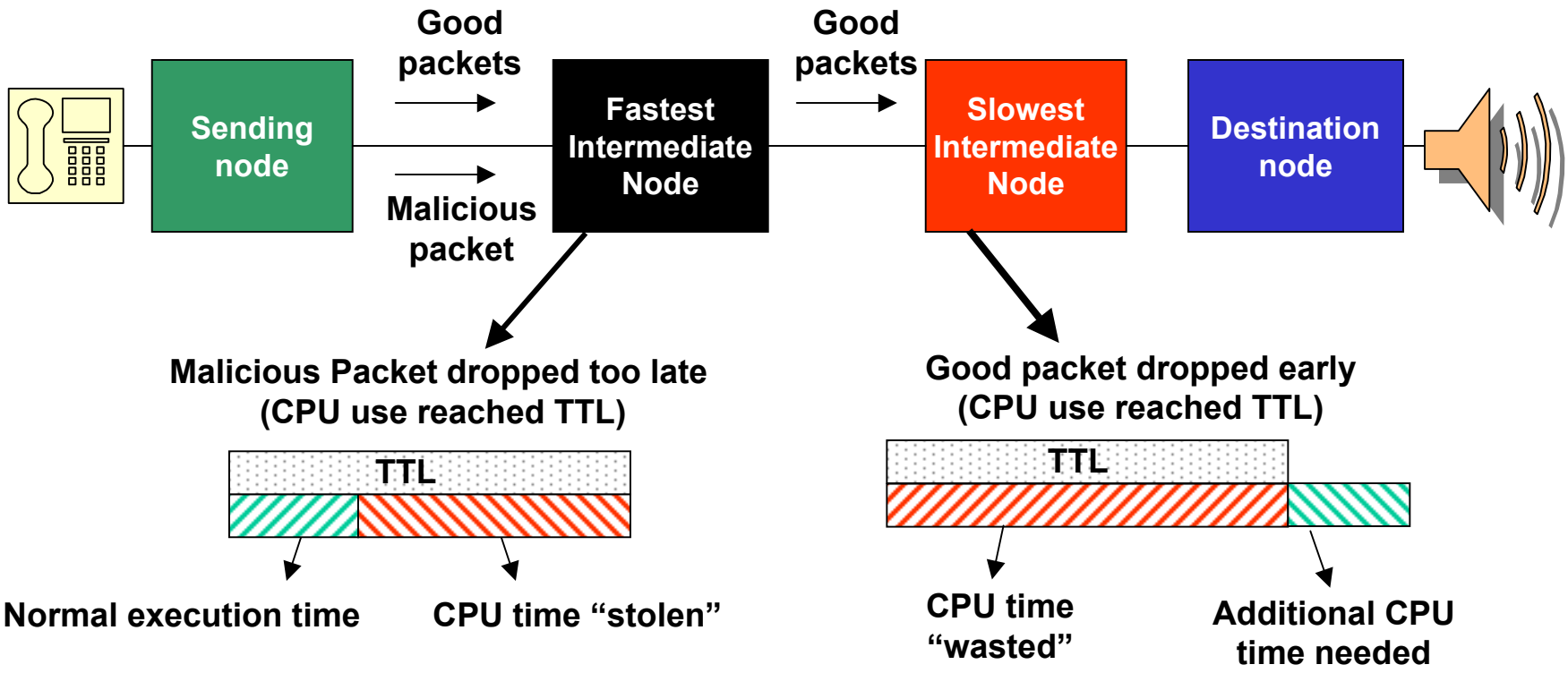
All nodes on the ABONE and running the Magician EE

Demonstration #1 Policy #1

Detect and Kill Malicious or Erroneous Packets

Demonstration compares three policies to enforce limits on CPU consumption

Policy 1: Use CPU time to live set to fixed value per packet

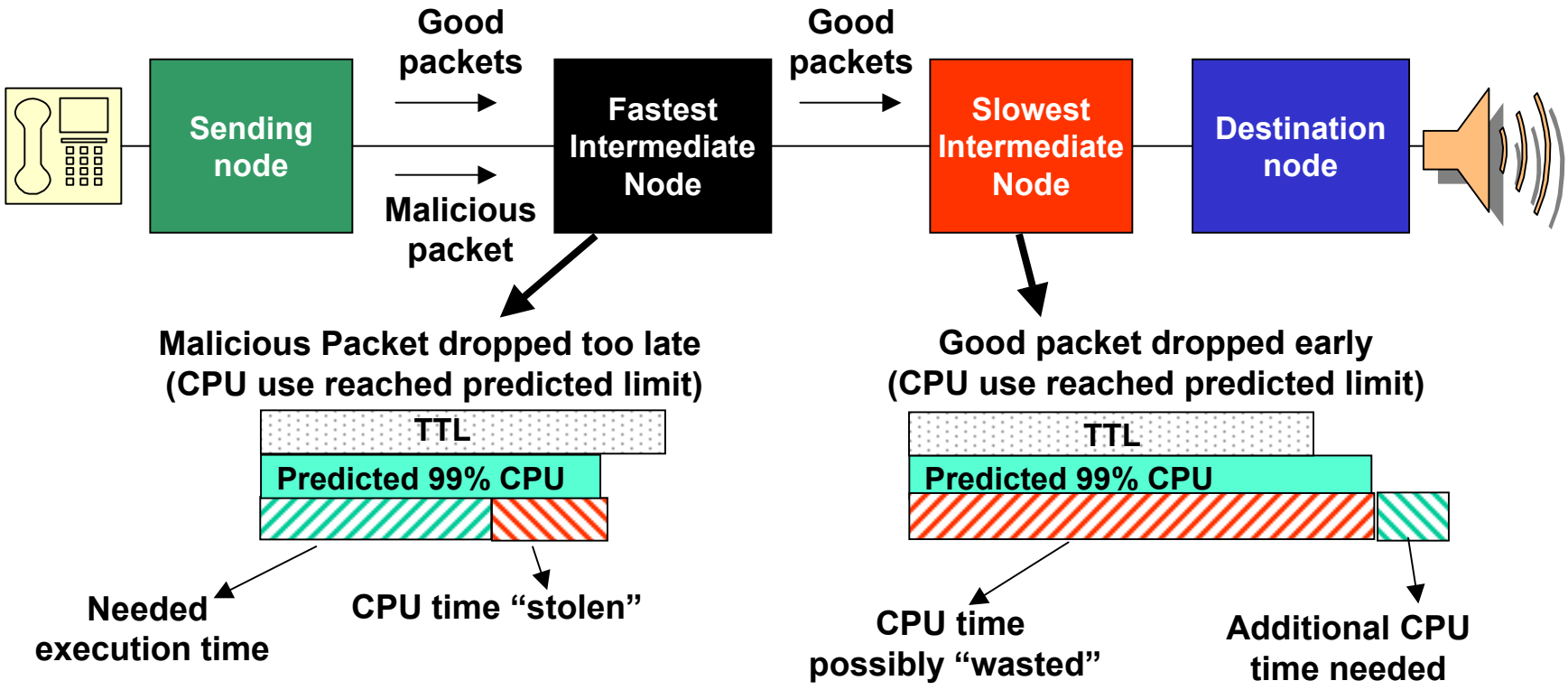


Demonstration #1 Policy #2

Detect and Kill Malicious or Erroneous Packets

Demonstration compares three policies to enforce limits on CPU consumption

Policy 2: Use a CPU usage model, but scaled naively based solely on CPU speed

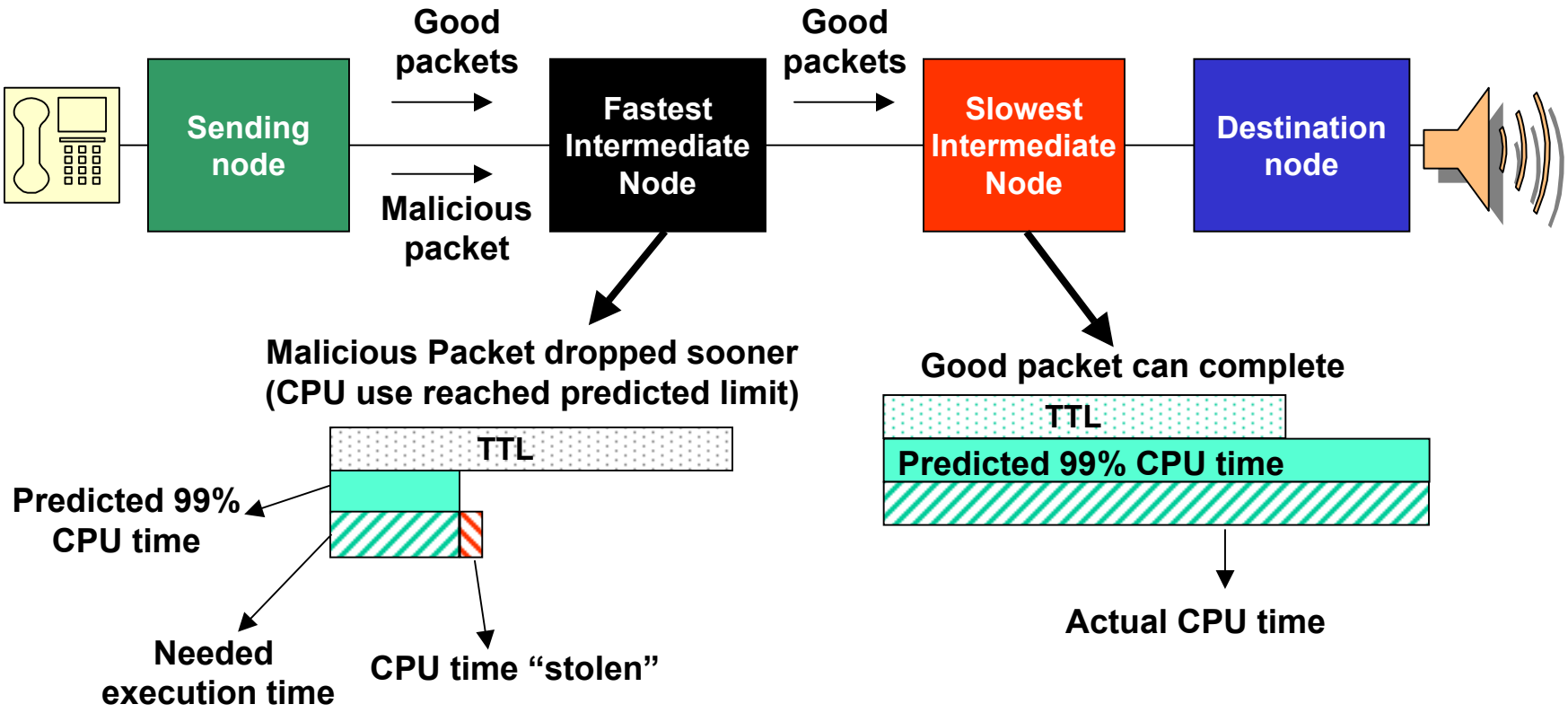


Demonstration #1 Policy #3

Detect and Kill Malicious or Erroneous Packets

Demonstration compares three policies to enforce limits on CPU consumption

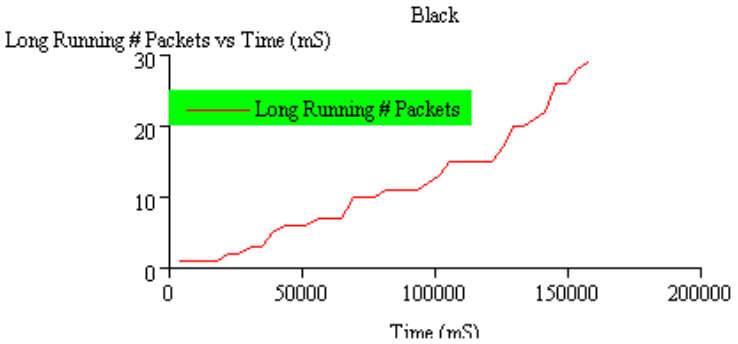
Policy 3: Use a well-scaled NIST CPU usage model

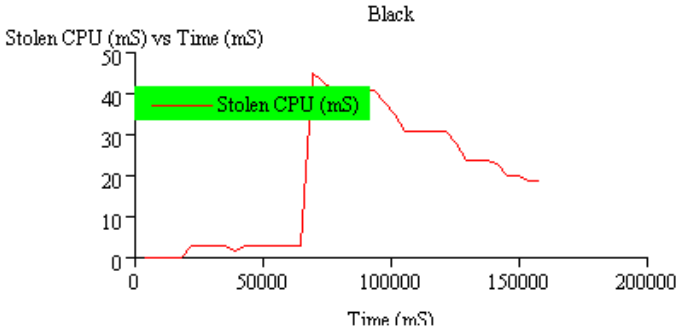


Summary of Demonstration #1

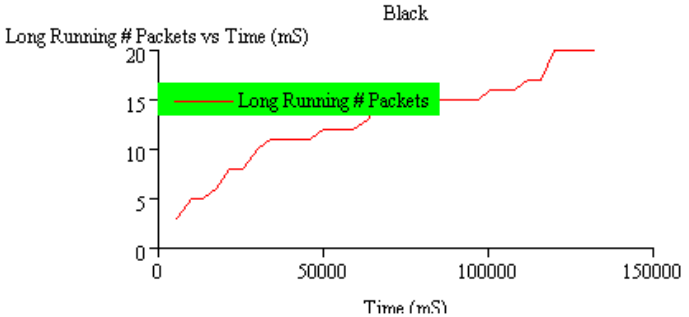
Detect and Kill Malicious or Erroneous Packets

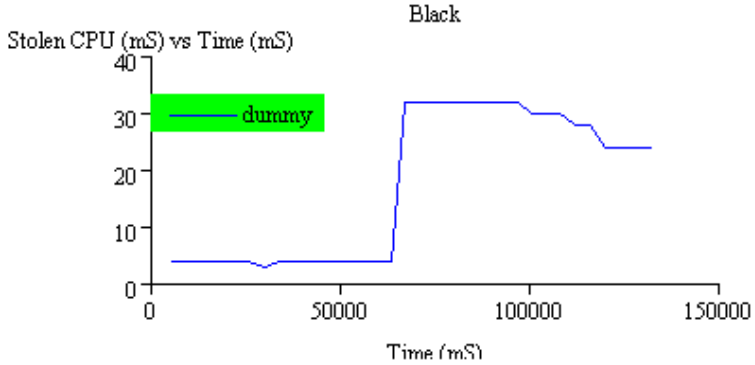
High Fidelity





Naïve Scaling





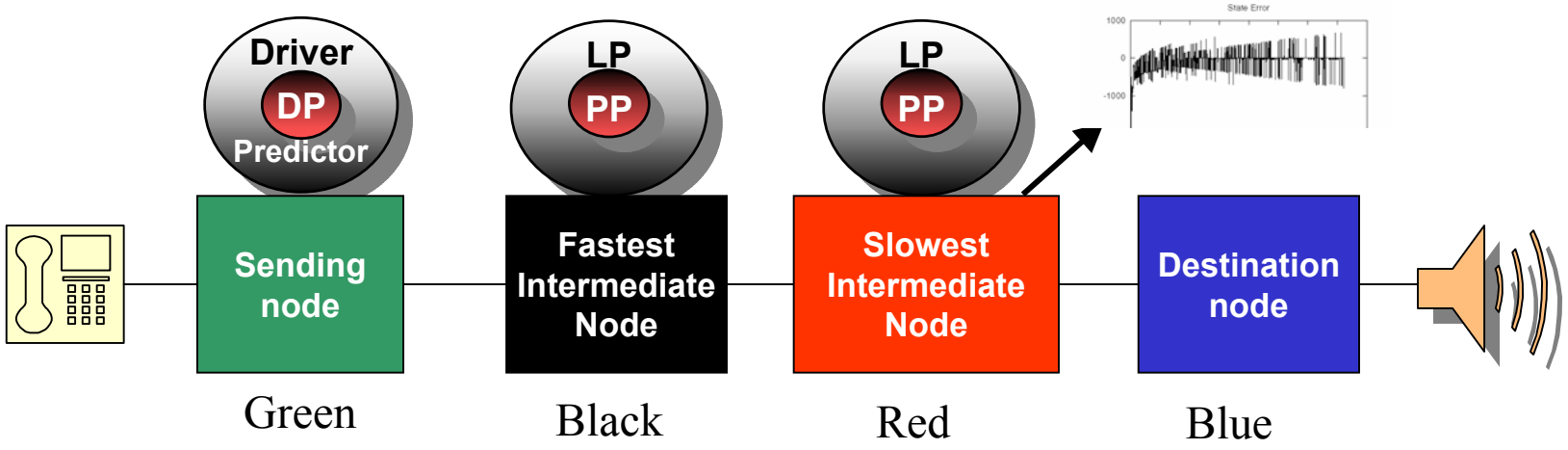
Demonstration #2 Overview

Predict Resource Usage, Including CPU Time, Throughout an Active Network

- Show that AVNMP can predict network-wide resource consumption
- Compare accuracy of AVNMP CPU usage predictions with and without the NIST CPU usage models
- Illustrate benefits when AVNMP provides more accurate predictions

Topology for Demonstration #2

Predict Resource Usage, Including CPU Time, Throughout an Active Network

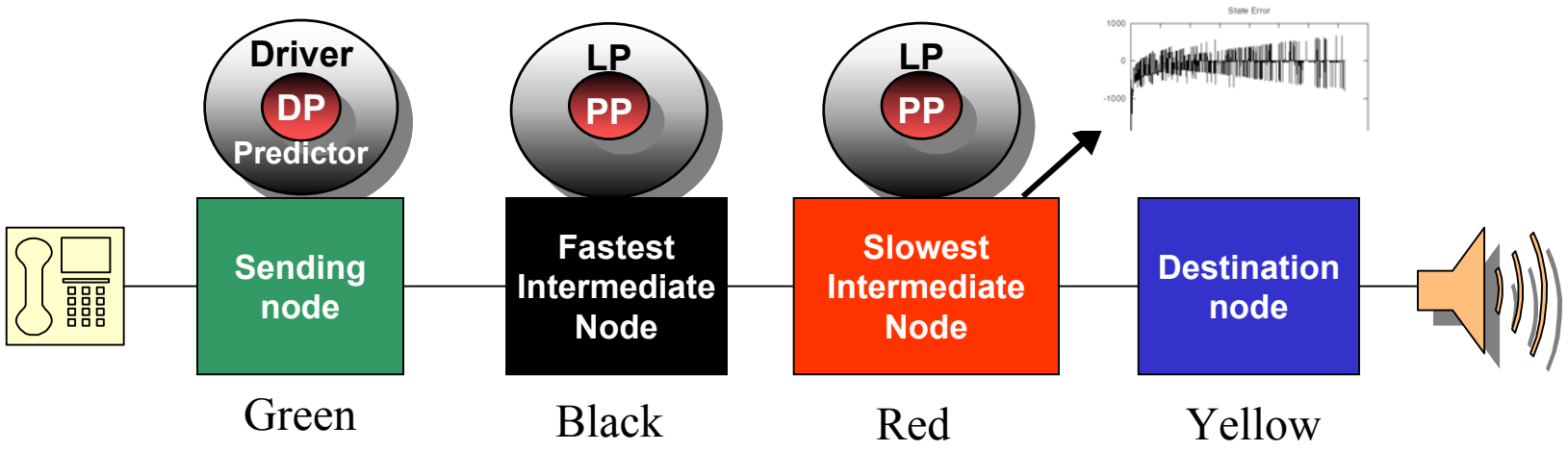


Demonstration #2

Predict Resource Use, Including CPU, Throughout an Active Network

Demonstrate predictive power of AVNMP and improvement in predictive power when combining NIST CPU usage models with AVNMP

With the NIST CPU usage model integrated, AVNMP requires fewer rollbacks

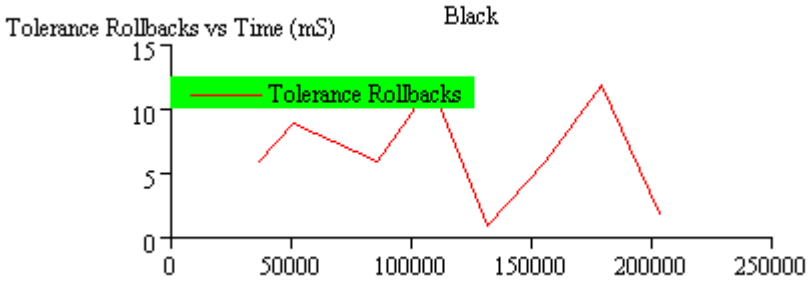


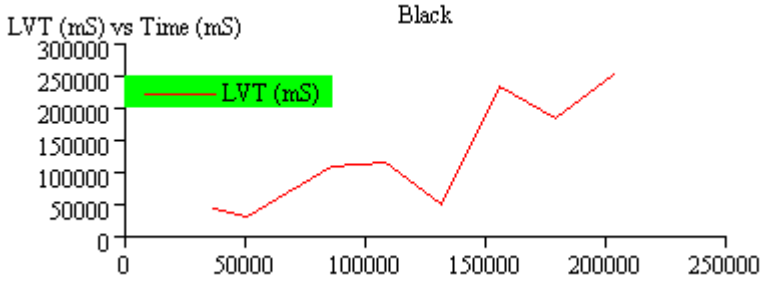
And so AVNMP can predict CPU usage in the network further into the future

Summary of Demonstration #2

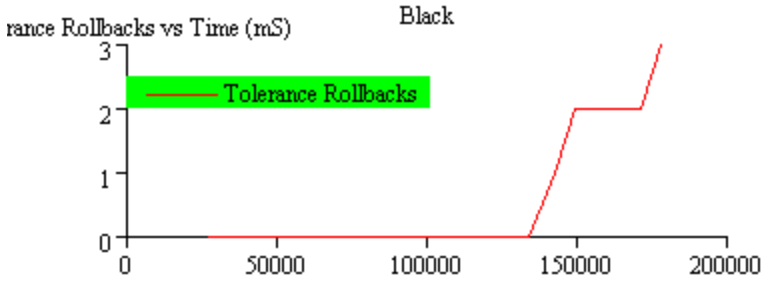
Predict Resource Use, Including CPU, Throughout an Active Network

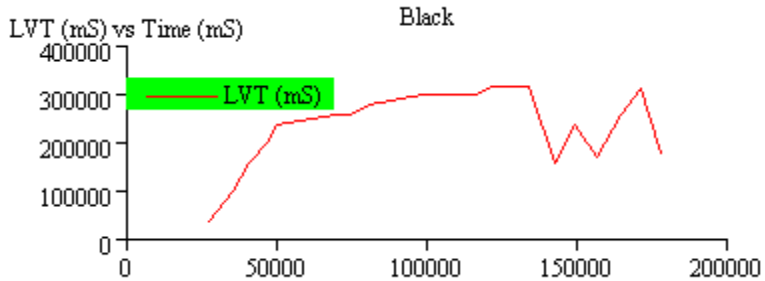
TTL





CPU Prediction





Better CPU prediction model overcomes performance tradeoff limitations

Accomplishments

- Demonstrated the ability to detect and kill malicious or erroneous active packets
  - Illustrated motivation behind CPU usage modeling
  - Compared three policies to enforce limits on CPU consumption
  - Showed improvement of NIST CPU usage models over naïve scaling
- Demonstrated management of CPU prediction and control of packets on per-application basis by an EE (Magician probably the first of its kind)
- Demonstrated the power of AVNMP to predict resource usage, including CPU, throughout an active network
  - Showed that AVNMP can predict network-wide resource consumption
  - Compared accuracy of AVNMP CPU usage predictions with and without the NIST CPU usage models
  - Illustrated benefits when AVNMP provides more accurate predictions
- Developed MIB for CPU and AVNMP Management of an active node
- Integrated SNMP agents and reporting in an EE
  - Provided user-customizable event reporting through multiple mechanisms: Event Logger and SNMP

Lessons Learned

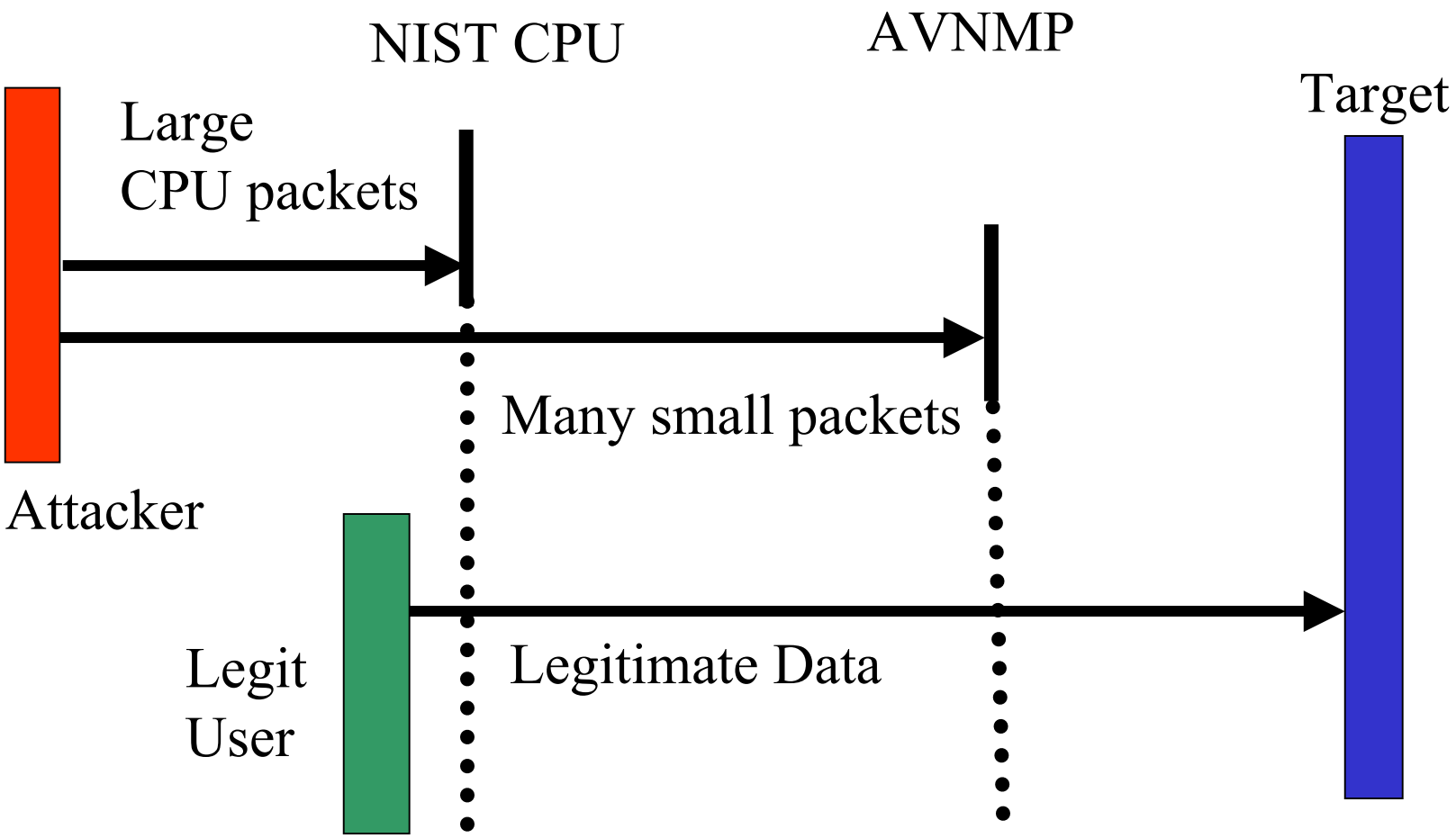
- **DO NOT KEEP MODIFYING** your demo code two days before the demonstration, especially when you are depending on detailed measurements of the code
  - Every AA change requires execution traces to be rerun
  - Every EE change requires execution traces **and** node calibrations to be rerun
  - In addition, new models must be generated for each platform
  - The good news – we were still able to do this
- NIST CPU benchmark tool should be made available in packaged form for rapid and easy use.
- Active Networks Architecture requires a standard interface for any EE to measure and control resource use by AAs
  - Working with two different EEs required these issues to be addressed uniquely for each EE
  - Using one technique to measure CPU use for AA model generation and another to measure CPU use in running AAs introduced unnecessary error
- Need to increase precision when CPU control mechanism terminates active packet (will Real-Time Java solve this?)
- Introduction of another roll-back variable suggests that AVNMP can prove even more efficient if roll-backs can be conducted independently on each class of variable

NIST Future Research

- Improve Our Models
  - Model Node-Dependent Conditions
  - Attempt to Characterize Errors Bounds
  - Improve the Space-Time Efficiency of Our Models
  - Continue Search for Low-Complexity Analytically Tractable Models
  - Investigate Models that Continue to Learn
- Investigate Competitive-Prediction Approaches
  - Run Competing Predictors for Each Application
  - Score Predictions from Each Model and Reinforce Good Predictors
  - Use Prediction from Best Scoring Model
- Apply Our Models
  - CPU Resource Allocation Control in Node OS
  - Network Path Selection Mechanisms that Consider CPU Requirements
  - CPU Resource Management Algorithms Distributed Across Nodes

Denial of Service Attacks

Can a combination of AVNMP load prediction and NIST CPU prediction be used to **combat denial of service attacks**?

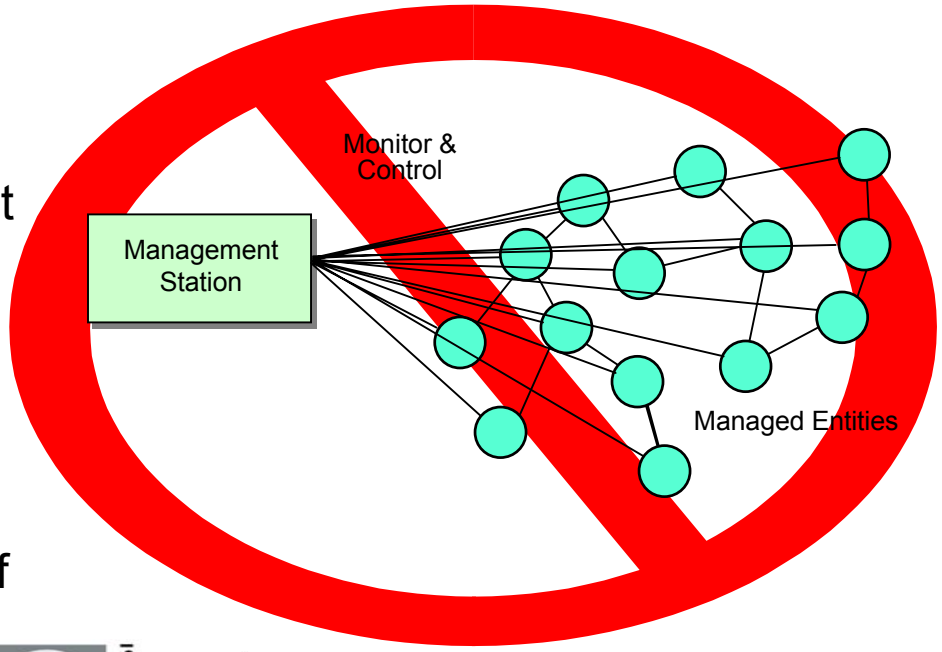


Goal: Large Networks with Inherent Management Capabilities

- Number of predicted objects will increase drastically -- many more than simply load and CPU -- see a typical SNMP MIB for possible number of predicted objects.
- Load and CPU have been demonstrated on a handful of nodes; but what about thousands of nodes and perhaps multiple futures?

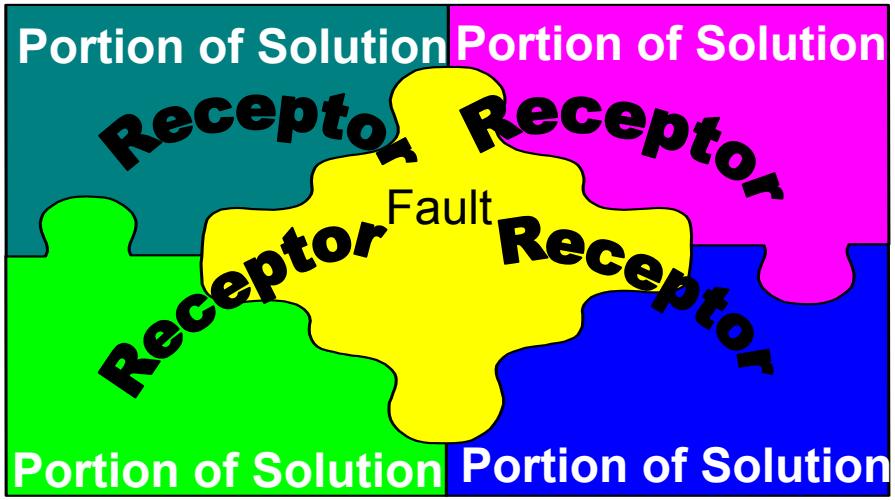
Today: Centralized, Manual, Brittle, External Management Systems

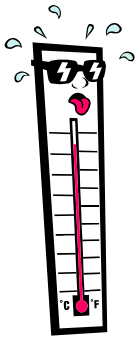
- Network management today is centralized…should be distributed
- Fault detection and correction are generally manual activities -- at best scripted…should be inherent to network behavior
- Unstable/Brittle…should be stable/ductile
- Management is external to the network…should be inherent part of the network



Network Inherently Forms Fault Corrective Action

- Identify faults within a complex system of management objects
- Scale in number of objects and number of futures
- Robust in the presence of faults
- Only necessary and sufficient repair capability should exist in time and space

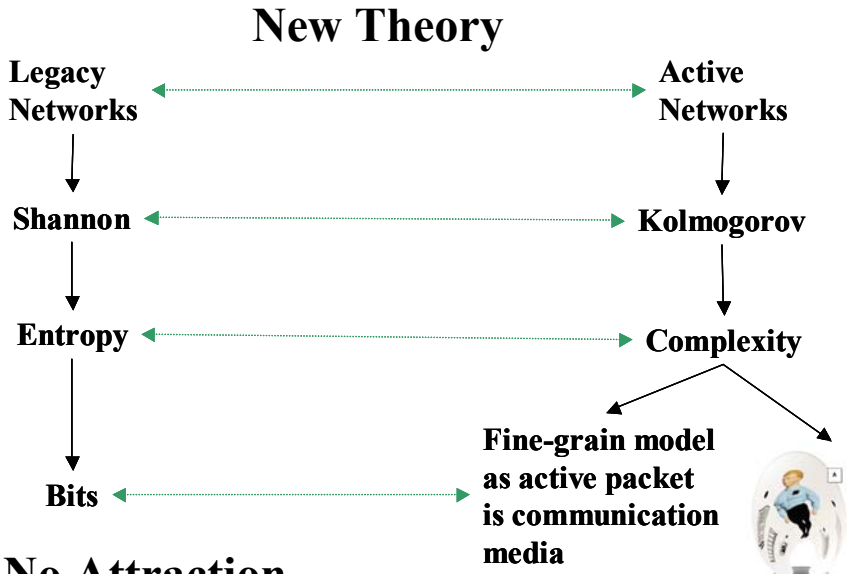


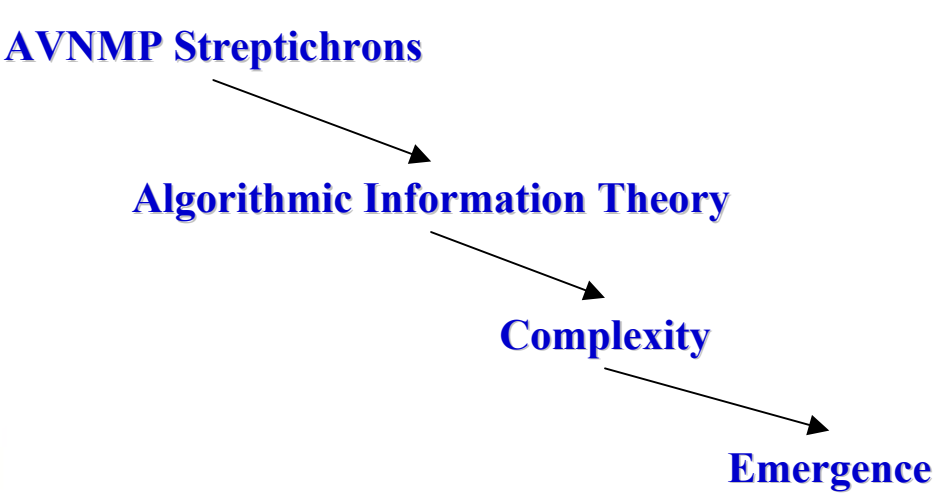


Exceeding "normal" ranges indicates a fault and generates <u>attractive force</u> needed to form corrective action.

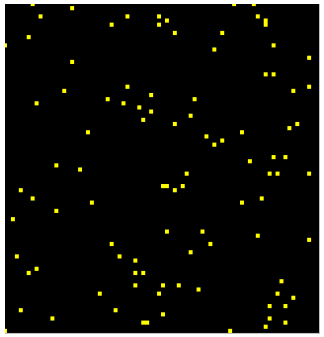
New Theory of Networks Leads

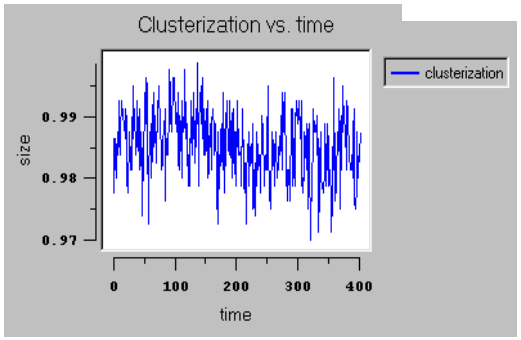
…to Example Applications such as Composition of State into Solution Attractors





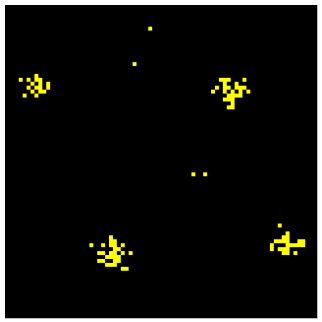
No Attraction

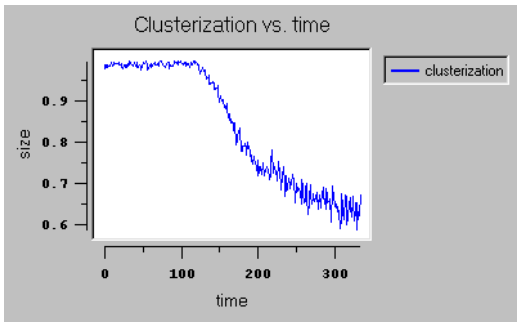




Random (Healthy) incompressible

Attraction





Order (Multiple Faults) compressible